Copyright

by

Rishi Alpesh Shah

2019

The Report Committee for Rishi Alpesh Shah
Certifies that this is the approved version of the following Report:

Deep R Learning for Continual Area Sweeping

APPROVED BY
SUPERVISING COMMITTEE:

Clinton N. Dawson, Supervisor

Peter Stone

Deep R Learning for Continual Area Sweeping

by

Rishi Alpesh Shah

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computational Science, Engineering, and Mathematics

The University of Texas at Austin

May 2019

Abstract

Deep R Learning for Continual Area Sweeping

Rishi Alpesh Shah, M.S.C.S.E.M

The University of Texas at Austin, 2019

Supervisor: Clinton N. Dawson

In order to maintain robustness, autonomous robots need to constantly update their knowledge of the environment, which can be expensive when they are deployed in large, dynamic spaces. The *continual area sweeping* task formalizes the problem of a robot continually patrolling an area in a non-uniform way in order to efficiently use travel time. However, the existing problem formulation makes strong assumptions about the environment, and to date only a sub-optimal greedy approach has been proposed. We generalize the continual area sweeping formulation to include fewer environmental constraints, and propose a novel reinforcement learning approach. We evaluate our approach in an abstract simulation and in a high fidelity Gazebo simulation, which shows significant improvement upon the initial approach in general settings.

Table of Contents

Deep R-Learning for Continual Area Sweeping

Rishi Shah* , **Yuqian Jiang*** , **Justin Hart** and **Peter Stone**
{rishihahs, jiangyuqian}@utexas.edu, {hart, pstone}@cs.utexas.edu

1 Introduction

Consider a service robot operating in an office or home. When a user requests for the robot to bring a cold beverage, or to pick up the mail, the robot must reason about not only the static facts - the locations of rooms, or the color of the mailbox - but also the locations of objects in its environment, which can change over time. As the activities of occupants of this environment constantly change the locations of objects, efficiently servicing these requests involves continually surveying the area for changes in objects and their locations.

The problem of *continual area sweeping* was introduced by Ahmadi and Stone [2005] as one motivated by building maintenance tasks in which some areas of the building see higher traffic and messier activities and therefore must receive more attention. Such a robot needs to service trash cans and restrooms more frequently than closets. The robot's utility is optimal when the time between the appearance of a mess and cleaning it up is minimal. They formalize the process of visiting areas of the map in a gridworld in which "events" (representing dirt and messes) appear non-uniformly throughout the environment. These events accumulate in grid cells until the robot enters into the grid cell to service them, and success is measured as how long this takes.

To model the task of a service robot surveying its environment for changes, this paper extends continual area sweeping. Instead of minimizing the time to service the biggest mess, our goal is to maximize the number of events detected per second. We also relax assumptions about the way events are distributed in the grid and accumulate over time, in order to better represent this scenario. We propose a novel solution using a Semi-Markov Decision Process in the average reward setting, in which the robot makes sequential decisions about where to travel to optimize long-term average number of detections per second. We then present a novel deep reinforcement learning approach, including a reward construction that reweights the immediate reward for this objective.

We evaluate this approach in two simulation domains. An abstract gridworld is used to analyze the effects of environmental assumptions and compare the performance of the Reinforcement Learning (RL) approach with the approach presented by Ahmadi and Stone, which serves as a baseline. Results show our RL approach significantly improves performance in the most general scenario. Our approach is then evaluated on a simulated service robot in Gazebo on the task of detecting object placements. We demonstrate that the approach is able to remember geometric features between different environments.

*equal contribution

2 Related Work

Coverage path planning, also known as terrain coverage or sweeping, is a problem in which, given a map of an environment, the goal of agents traversing that map is to generate a path such that the the agent passes through the entire volume of the map [Choset, 2001; Gabriely and Rimon, 2001; Galceran and Carreras, 2013]. This approach is useful for a variety of applications where the robot must travel over the entire area, such as lawn mowing or vacuum cleaning. Significant work has also been done on sweeping in multi-robot settings, [Kurabayashi *et al.*, 1996; Zlot *et al.*, 2002], where the problem can be expressed as one of efficiently the path planning or traversal behavior of multiple robots.

Ahmadi and Stone [2005] introduced the continual area sweeping problem as a specialization of this problem (and followed-up with a multi-robot case [Ahmadi and Stone, 2006]) where the goal of the agent is to provide greater coverage to areas where more activity occurs. Their goal is similar to that of the cleaning robot from above, but enabling it to autonomously focus on high-traffic areas where a greater amount of mess accumulates. The problem is modeled over a grid, where an event occurs in each grid cell according to a random distribution. They also present an initial approach to this problem; a greedy approach to minimizing the average time it takes to detect an event once it has occurred.

The motivation for this work is to enable autonomous service robots to maintain a high degree of awareness of their environment. We approach this problem as one related to semantic mapping problems, of which there are a variety. Nuchter et al [2006] labeled detected terrain in 3D maps as belonging to entities such as floors or ceilings. Hart et al [2018] incorporate room number information into their map so their robot can navigate to specific rooms. Mason et al [2012] use perceptual data to discuss objects in the scene. Important in their system is change detection, where semantics are updated when they have changed in the environment. Similarly, SOMA concentrates on the exploration of objects and change over time [Kunze *et al.*, 2018]. Robust, autonomous exploration of map data is possible through frontier-based exploration methods [Yamauchi, 1997], though we have not yet seen such a complete search expanded to semantic mapping. However, Jebari et al [2011] develop a stochastic method for autonomous exploration of an area in 2D using a Lidar unit, then scanning for objects using a pan-tilt camera. The present work is a step toward a continual scan of an area for the purpose of updating information in a semantic map. Our eventual goal is to excel in maintaining awareness of the scene, keeping an up-to-date semantic map by using RL to inform the robot of good exploration strategies that are likely to provide information updates.

3 Problem Formulation

In the continual area sweeping task, a robot continually travels in an environment with the goal of detecting or reacting to events of interest. The environment is represented as a 2D map which is divided into a set of discrete grid cells $g \in G$. A set of events $e \in E$ can occur anywhere in the environment at any time t. The robot makes sequential decisions which are broken down into discrete decision steps $n \in \mathbb{N}$. At each decision step n, the robot can take an action a_n to move to any reachable grid cell g (including staying at the current cell). This action space focuses on the decision of where to visit, and abstracts away path planning in specific domains. When an action is executed, the robot is able to detect any events in every grid cell along its path. The number of such detections is d_n. Note that the robot must physically travel from grid cell to grid cell, and as such may take a variable length of time to do so. As such, we denote the wall-clock time of decision step n as t_n.

3.1 Metrics

We define two metrics, *average detection time (ADT)* and *detections per second (DPS)*, each appropriate for a different category of applications.

The *average detection time (ADT)* is the average time elapsed from occurrence to detection of the events. More formally, let $a(e)$ denote the time when event e occurs in the grid, and let $b(e)$ denote the time at which event e was detected. If e has never been detected, then let $b(e)$ be the current time. Then ADT is precisely $\frac{1}{m}\sum_{i=1}^{m}(b(e_i) - a(e_i))$, where m is the total number of events that have occurred. This metric is used in the original continual area sweeping formulation [Ahmadi and Stone, 2005]. If the goal of the robot is to be highly responsive to emergencies, such as a spilled drinks for a maintenance robot, then it is appropriate to optimize average detection time.

Detections per second (DPS) on the other hand, is the average of the number of events detected per unit time, computed as $\frac{1}{t_n}\sum_{i=1}^{n} d_i$. If the goal of the robot is to maintain up-to-date information in its environment, then it should detect as many changes as possible over time. Thus, maximizing detections per second is more meaningful.

Note that both metrics are defined in the continual setting, so we care about the long term average as m and n become arbitrarily large.

3.2 Assumptions

We assume that at each time step, the number of events in a grid cell $g \in G$ has an upper bound. In the cleaning task, an event can be described by a boolean variable of whether there is trash in a grid. Events can also stop after they occur. In the object tracking case, if a water bottle is placed on a desk, and its owner later picks it up, the event of the disappearance of the object overwrites the event of its appearance. In realistic domains, the bound on number of events in each grid cell is usually close to 1 for a fine-enough grid representation.

4 Approach

In this section, we first explain the baseline ADT-Greedy algorithm, a prior approach from the literature. Next, we model our formulation of continual area sweeping as a Semi-Markov Decision Process that we then use to introduce a novel deep RL approach called DPS-Max which provably maximizes average DPS.

4.1 Baseline ADT-Greedy Algorithm

The initial continual area sweeping approach by Ahmadi and Stone [2005] greedily optimizes for the average detection time (ADT). We refer to this approach as *ADT-Greedy* in this paper. ADT-Greedy makes the assumption that, at each time step, there is a fixed probability p_g for an event to occur at a grid cell g. Therefore, the number of events in each grid cell follows a binomial distribution $B(t, p_g)$, where t is the number of time steps since the cell was last visited. We call this assumption the *binomial assumption* in this paper. Moreover, ADT-Greedy assumes there is no upper bound on the number of events per cell, which we call the *unbounded events assumption*. The effects of violating the binomial assumption and the unbounded events assumption are analyzed in Section 5.1.

A convenient consequence of the assumptions is that the expected number of events in a cell is linear in the time since the cell is last visited. This is because the expectation with respect to the Binomial distribution $B(t, p_g)$ is $t \cdot p_g$. Ahmadi and Stone show that maximizing the total expected number of events is the same as minimizing ADT.

The ADT-Greedy algorithm consists of a learning module that learns p_g for every grid cell g. A planning module then greedily chooses the target cell that leads to the path with the highest expected number of events. The algorithm thus offers a greedy approach to minimizing ADT.

4.2 Semi-MDP Model

We now describe our proposed DPS-Max, starting with how it models the continual area sweeping problem as a Semi-Markov Decision Process. A Semi-Markov Decision Process consists of $(\mathcal{S}, \mathcal{A}, R, P)$:

- $\mathcal{S}$ is the state space. Each state includes G and the position of the robot $g \in G$. Also, for every event, we have an exponential decay of the time since the event was last seen, stated as $\exp(-\alpha t_d)$, where t_d is time since an event was last detected and α regulates the rate of decay.

  These components are stored as grids in order to represent the topology of the map in our state space. Specifically, the robot's position is encoded as a one-hot grid, and the exponential decays are stored in a grid at the cell corresponding to their events. Combined with a neural network function approximator, this grid representation encodes the fact that local spatial regions of the state should be similar.

- $\mathcal{A}$ is the action space, which includes all positions in G. The robot takes one action a_n at each decision step n. The key difference between an SMDP and an MDP is that these decision steps do not need to correspond to wall-clock time in the environment. A consequence is that actions all take one decision step, but can have different execution times in the actual environment. This is because actions represent motion from the grid cell occupied by the robot to any grid cell in the map, so the time an action takes to execute is proportional to the length of the robot's path.

- P is the transition kernel, which is unknown to the robot.

- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a measurable function denoting the reward given for a transition.

A stationary policy π describes the action to take in a given state, and is thus a map from $\mathcal{S}$ to probability measures on $\mathcal{A}$.

Since continual area sweeping operates in a continual setting, the average reward formulation is the best suited setting. ρ^π is the average reward function:

$$\rho^\pi(\mu) \coloneqq \liminf_{n\to\infty} \frac{1}{n} \mathbb{E}\Big[\sum_{k=0}^{n-1} R(s_k, a_k, s_{k+1})\Big] \tag{1}$$

where μ is an initial state distribution, and the expectation is taken with respect to the appropriate measure derived from π and μ [Feinberg, 1996]. For convenience, when $\mu(s) = 1$ for some state s, we use the notation $\rho^\pi(s)$.

This leads to the optimal differential value function:

$$Q^*(s,a) = \mathbb{E}_P R(s,a,s') - \sup_\pi \rho^\pi(s) + \mathbb{E}_P\big[\max_{a' \in \mathcal{A}} Q^*(s', a')\big]$$

The goal under this SMDP formulation is to approximate Q^*.

4.3 Reward Construction

The reward function of DPS-Max is formulated to maximize average detections per second. It is tempting to assume that the reward function can be defined as this rate itself–after all, if average reward is maximized, then setting the reward to the rate should maximize average rate. The problem is that SMDP decision steps do not correspond to wall-clock time, so the average reward setting averages over SMDP decision steps, while our rate is in terms of time. Thus, setting the reward function to be a rate will only lead to maximizing the rate itself when actions take the same amount of time; otherwise, special care is needed.

Reward construction is an important part of SMDP design, and many schemes deal with handling the time and decision step mismatch [Baykal-Gürsoy, 2010]. Here, we design a reward function specifically for the case of optimizing a rate, such as maximum detections per second.

Proposition 1. *Take $\{(s_n, a_n)\}_{n\geq 0} \subset \mathcal{S} \times \mathcal{A}$ to be a trajectory generated from a policy π. Let $\{\phi_n\}_{n\geq 0} \subset \mathbb{R}$ a sequence, and $\{t_n\}_{n\geq 0} \subset \mathbb{R}$ an increasing sequence denoting the associated environmental time. Construct R in the following way:*

$$R(s_0, a_0, s_1) \coloneqq 0$$
$$R(s_n, a_n, s_{n+1}) \coloneqq (n+1)\frac{\phi_{n+1}}{t_{n+1}} - n\frac{\phi_n}{t_n}$$

Then $\rho^\pi(s_0) = \liminf_{n\to\infty} \frac{\mathbb{E}\,\phi(s_n)}{t_n}$

Proof.
Substituting in (1):

$$\rho^\pi(s_0) = \liminf_{n\to\infty} \frac{1}{n} \mathbb{E}\Big[\sum_{k=0}^{n-1} (k+1)\frac{\phi_{k+1}}{t_{k+1}} - k\frac{\phi_k}{t_k}\Big]$$

The sum telescopes out leading to:

$$\rho^\pi(s_0) = \liminf_{n\to\infty} \frac{1}{n} n \frac{\mathbb{E}\,\phi_n}{t_n}$$

□

The corollary of this proposition is that by setting ϕ_n to be the number of detections seen at step n, we are provably optimizing average detections per second.

Algorithm 1 Deep R-Learning

1: Initialize empty experience replay buffer $\mathcal{D}$.
2: Initialize network Q with random weights $\boldsymbol{\theta} = \boldsymbol{\theta}^{-}$.
3: Initialize $\rho = 0$.
4: **for** t = 1, . . . , M **do**
5: Select an action a_t according to an action selection mechanism like ϵ-greedy.
6: Execute a_t and store the resulting transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$.
7: Randomly sample a batch of transitions $\{(s_j, a_j, r_j, s_{j+1})\}$ from $\mathcal{D}$.
8: Let $q_{max} = Q\left(s_{j+1}, \text{argmax}_a\, Q\left(s_{j+1}, a; \boldsymbol{\theta}\right); \boldsymbol{\theta}^{-}\right)$.
9: Let $y_j = r_j - \rho + q_{max}$.
10: Take a gradient descent step on $L\big(y_j, Q\left(s_j, a_j; \boldsymbol{\theta}\right)\big)$.
11: Let $\Delta_j = y_j - Q\left(s_j, a_j; \boldsymbol{\theta}\right)$
12: Let $\Delta = \text{avg}\{\Delta_j \text{ s.t. } |Q(s_j, a_j) - q_{max}| < \delta\}$
13: **if** Δ is well-defined **then**
14: $\rho = \rho + \alpha\Delta$ for learning rate α
15: **end if**
16: **end for**

4.4 Deep R-Learning

R-Learning is a classical approach for learning an optimal differential value function [Schwartz, 1993]. Its purpose is in handling infinite-horizon tasks where finding a policy that maximizes average reward is more meaningful than temporal discounting. For this problem, discounting is not a good fit as it is a continuing problem where we wish to optimize an average rate (DPS). Although the theory is less developed than that of discounted algorithms, R-Learning has been shown to be useful empirically. However, a suitable function approximator is needed to represent the value function. To that end, we introduce a deep variant of R-Learning. It is based on double DQN [Van Hasselt *et al.*, 2016], which allows for the integration of neural networks with double Q-Learning.

Algorithm 1 describes our algorithm. The key changes to double DQN are highlighted here. First, the target in line 9 reflects the R-Learning update by subtracting out the running average reward estimate. Lines 11 and 12 compute the change to ρ. Here, the TD errors of the batch are averaged so long as the actions taken were close to optimal. As a result δ essentially controls a bias-variance trade off of average reward updates. A low δ will lead to lower bias as it is closer to approximating ρ^{π^*}, but there will be higher variance as it takes smaller batch averages. If line 12 attempts to take the average of an empty set, then the subsequent if-statement will not execute.

4.5 Q Function Representation

To represent Q in Algorithm 1, we use an autoencoder network as a way of exploiting the topology of $\mathcal{S}$ and $\mathcal{A}$. For a practical map, there can be millions of actions, since the agent can choose to move anywhere (resulting in close to height$\times$width of G number of actions). Value based methods are normally poorly suited for such a large action space, but this choice of architecture overcomes that limitation. Due to convolutional layers, updates made to the Q-value of a state-action pair immediately generalize to a local neighborhood. Figure 1 illustrates the architecture[1]. The max-pool and upsampling layers help coalesce the action values of neighbors.

[1] In our experiments, the architecture comprises a $32\times5\times5@3$ conv, $2\times2@2$ max-pool, two $16\times4\times4@2$ conv, and a 500 unit fully connected layer, while the decoder part is the reverse with the conv layers swapped for deconv layers and upsampling for the max-pool.

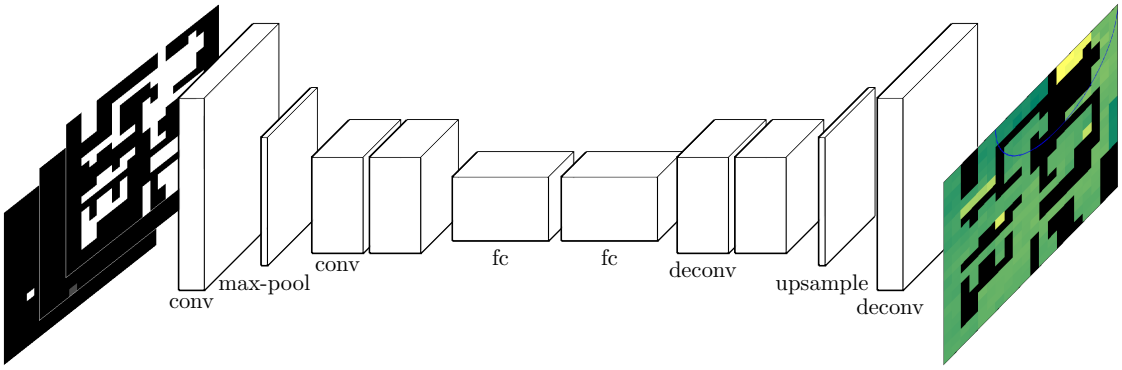


Figure 1: Autoencoder network where the environment map, robot position, and time information about seen objects are represented as grids and fed in as the input. The output is the action-value for each cell (action) in the map.

5 Experiments

This evaluation employs two domains to evaluate our approach: an abstract gridworld, and a simulated house in Gazebo. The gridworld is used to compare the performances of the *DPS-Max* approach and the *ADT-Greedy* approach under different environmental assumptions. The RL approach is further implemented on a simulated domestic service robot in the Gazebo robotics simulator in order to evaluate its ability to continually detect semantic changes in the environment.

5.1 Effects of Assumptions

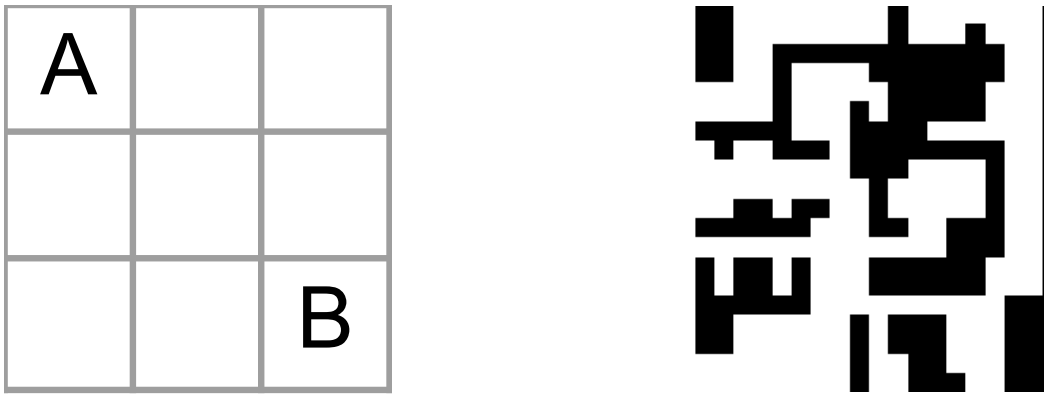
We present examples in a small gridworld to illustrate how the *binomial assumption* and the *unbounded events assumption* affect the performance of continual area sweeping policies.

Binomial Assumption

Consider the 3×3 gridworld in Figure 2a where events can occur at grid cell A and B. The robot's initial position is at A. Let X_A and X_B be the number of events that occurred in grid cells A and B before time t. Under the binomial assumption, there are fixed probabilities p_A and p_B such that $X_A \sim B(t, p_A)$ and $X_B \sim B(t, p_B)$.

Suppose the assumption holds, and $p_A = 1/2$, $p_B = 1/4$. By the ADT-Greedy algorithm, the first 9 actions are (A, A, A, A, A, B, A, A, B), and then the actions repeat in the cycle of (A, A, B). After 100,000 actions, the ADT is 3.41, and the DPS is 0.75.

Suppose the events in A and B, instead of occurring with fixed probabilities, always occur periodically every 2 seconds and every 4 seconds, respectively. ADT-Greedy will learn the same p_A and p_B as the binomial setting, and have the same performance on both metrics since the events still appear at the same rate. However, the policy can be improved by exploiting the periodic pattern of events. For example, consider a simple policy that repeats in the cycle of (B, A). The robot starts in A, arrives in B at timestep 4 when the first event in B appears, comes back to A, and goes to B again. Since the distance between A and B is 4, the timestep is always a multiple of 4 when the robot is in B, and a multiple of 2 when the robot is in A. After 100,000 actions using this custom policy, the ADT is 2.67s, and the DPS is 0.75. Therefore, ADT-Greedy can be outperformed in ADT when the binomial assumption is



(a) (b)

Figure 2: (a) Gridworld where events occur at grid A and B. (b) Gridworld where black represents walls. In 5 random cells, events have a periodicity ranging from 10 to 50 seconds in the periodic case, or show up with a fixed probability ranging from $1/10$ to $1/50$ per second in the binomial case.

violated. In service robot domains, many events are naturally periodic, such as food that appears in dining areas at roughly the same time everyday.

Unbounded Events Assumption

The effects of violating the unbounded events assumption can be demonstrated in the same gridworld in Figure 2a. Suppose the binomial assumption holds with $p_A = 1$, $p_B = 1/2$. ADT-Greedy produces the same action sequence: (A, A, A, A, A, B, A, A, B) at first and repeating thereafter in the cycle of (A, A, B). Since ADT-Greedy makes the unbounded events assumption that an unbounded number of events can accumulate in B, it chooses to go to B frequently. In reality, the other extreme usually holds: each cell can have at most one active event at a time. For instance, if the event is the appearance/disappearance of a specific object, then each new event overwrites the last one. In this case, it is better to always stay in A than frequently visiting B. After running the action selection algorithm of ADT-Greedy for 100,000 actions, the ADT is 4.67s, and the DPS is 0.33. Alternatively, if the robot stays in A for 100,000 actions, the ADT is 1.00s, and the DPS is 1.00, leading to better performance on both metrics.

5.2 Gridworld Experiments

This evaluation tests on a larger gridworld in order to demonstrate the performance of the proposed algorithm under the binomial assumption and the unbounded events assumption.

Setup

Figure 2b illustrates the setup for the following gridworld experiments. A 20×20 grid is populated with random locations at which events may occur. Events appear under the binomial assumption, or occur periodically. In the binomial case, events appear with a fixed probability between $1/10 - 1/50$ each time step, and in the periodic case, according to a fixed period between $10 - 50$ time steps. These events occur in 1 of 5 fixed locations which are randomly generated at the start of each experiment, with a probability or time period associated with each of the 5 locations at the start of the experiment. We also evaluate the effects of the bound on the number of events per grid cell by varying the bound from 6 to 1. This tests the effect of the binomial assumption made by the ADT-Greedy algorithm, with 6 being closer to the original assumption and 1 completely violating it.

For each configuration, 8 grids of random object positions and occurrence probabilities/periods are generated. Since the instances have randomly generated event patterns, the best achievable DPS and ADT are different for each

instance. Therefore, we compare DPS-Max to ADT-Greedy by taking the percentage difference in our DPS or ADT over the DPS or ADT of ADT-Greedy, averaged across each configuration.

In this set of experiments, the learning rate α in Algorithm 1 is set to 0.0001. The exploration strategy is to initialize the agent in a random position, run ϵ-greedy exploration for 50 steps, and then reset the agent to a random position. The low learning rate and the frequent resets are used to ensure sufficient exploration.[2] The stopping criterion for training is the following: after every 20,000 training steps, the model is evaluated by executing the policy at a random initial position, and training terminates if the DPS has not improved in the last 10 roll-outs.[3]

Results

Figure 3a shows the average percentage difference in DPS, where higher than 0 means DPS-Max detects more events per unit time than ADT-Greedy. Figure 3b shows the average percentage difference in ADT, where lower than 0 means on average DPS-Max takes less time between event appearance and detection than ADT-Greedy. The error bars in the figures report standard deviation.

As shown by both figures, DPS-Max has the most advantage over the ADT-Greedy approach when the binomial and unbounded events assumptions are most violated. When event appearance is periodic and the number of events in each grid is bounded by 1, DPS-Max achieves the best improvement in DPS (43.7%) and the most reduction in ADT (38.4%). The reduction in ADT is surprising because unlike ADT-Greedy, DPS-Max does not directly optimize for ADT. In fact, the two metrics align except for a few cases. For instance, when the bound is 4 in the periodic setting, DPS-Max has better DPS but worse ADT compared to ADT-Greedy.

When the bound on events in every grid is high, DPS-Max does not outperform ADT-Greedy on either metric. One possible explanation is that when visiting a grid that has many active events, there is a large reward which causes instability in learning. Such a scenario is not the focus of this work, and we leave further investigation of this case to future work.

5.3 Gazebo Simulation

Gazebo is a high fidelity robot simulator [Koenig and Howard, 2004] that we use to simulate the Toyota Human Support robot in an indoor environment. We conduct the following set of experiments in this setting as it presents a realistic simulation of a robot. First, actions taken in Gazebo are noisy; the same action can take varying amounts of time to execute, and actions sometimes fail, causing the robot to stop midway through. Moreover, the environment map is large (300x300 grid representing a 900 square meter area). In short, successful learning in Gazebo requires sufficient robustness and generalization from the learning algorithm.

Remembering Geometric Features

A desirable feature of a continual area sweeping agent is the ability to remember geometric features. If objects are regularly placed on some piece of furniture, and the robot sees the furniture move during training, then the robot should naturally remember how to handle the moving piece of furniture. DPS-Max can accomplish this due to its convolutional network's ability to extract local features from a grid-based state. In contrast, ADT-Greedy by design forgets about learned events. This is necessary as there is neither a feature-based representation of the map nor a feature extractor, so new information about grid cells must override old information.

[2] Otherwise, exploration tends to stick around grid cells with frequent events, and not cover enough of the state space; causing high variances in evaluations since the robot's initial position is random.

[3] On average, training terminates around 400,000 steps in this set of experiments.

Setup

The robot is placed in an empty room with a cubicle, pictured in Figure 4. The robot is trained off-policy where data is gathered by having the robot navigate through a human generated path roughly covering the whole room. During this training period, the robot sees the cubicle from different positions. As we are mostly interested in the navigational aspects of the problem, we bypass robot perception and instead classify an event as having been detected if the robot is within 2 meters.

Events dependent on furniture

In our first test, an event constantly appears in the cubicle. This cubicle is moved around, and the robot experiences this change in position during training. There is also a second event continually firing at a fixed position. After training, the learned policy is run with the cubicle in two positions that the robot had seen earlier during training. The result is shown in the top row of Figure 5. The fact that the *same* policy produced tailored paths depending on the location of the cubicle shows that the robot was able to associate the visual appearance of the cubicle with the fact that events often appear there.

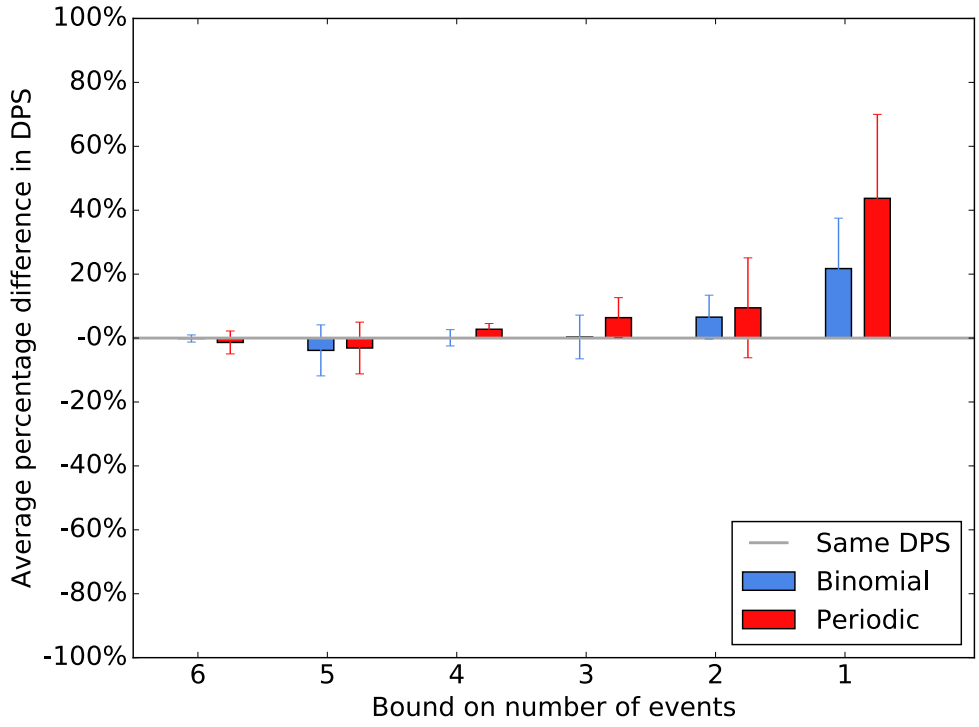
Events independent of furniture

Remembering geometric features is not useful if the robot constructs false associations. If the cubicle moves, but events do not move with the cubicle, then the robot should simply ignore the geometry of the cubicle. To test this ability, we repeat the previous experiment, but with both objects fixed. The same learned policy has the robot ignore the moving cubicles as seen in the bottom row of Figure 5.

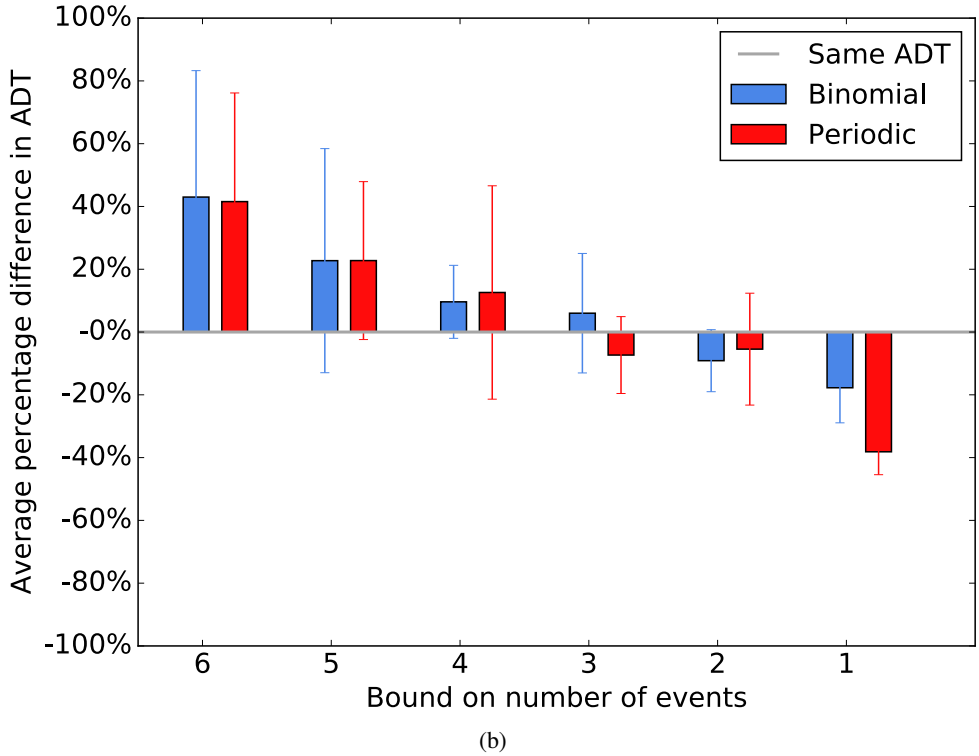
These two experiments show that the robot is able to memorize geometric features and recall them on demand, but only when they are truly relevant.

6 Conclusion

In this work, we extend the formulation of the continual area sweeping problem using an SMDP, and propose a deep R-learning approach to maximize average detections per second. These developments lead to an improvement upon the initial baseline, especially under more relaxed assumptions, and allow for the memorization of geometric features. In future work, we hope to apply and test this approach on real service robots as an idle sweeping behavior that improves knowledge of their environment.



(a)



(b)

Figure 3: (a) Average percentage difference in detections per second (DPS) of DPS-Max over ADT-Greedy. (b) Average percentage difference in average detection time (ADT) of DPS-Max over ADT-Greedy.

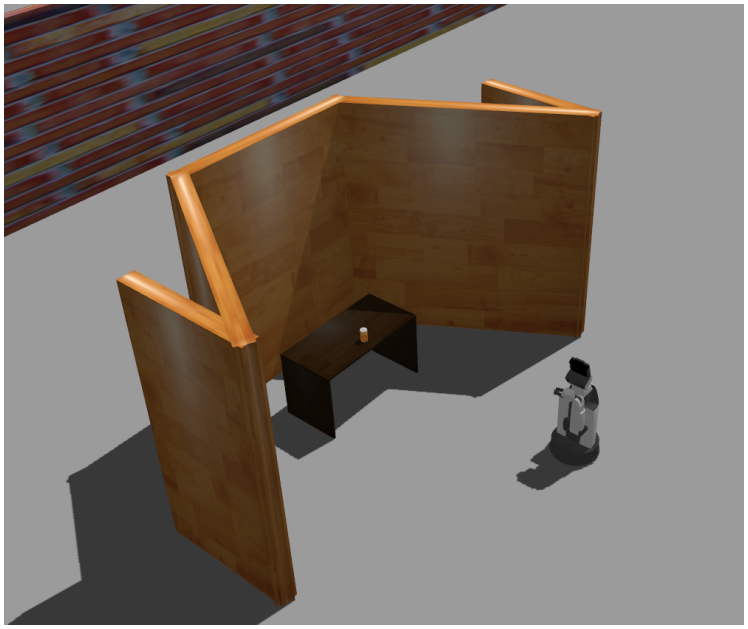


Figure 4: Cubicle and the robot in our simulated room.

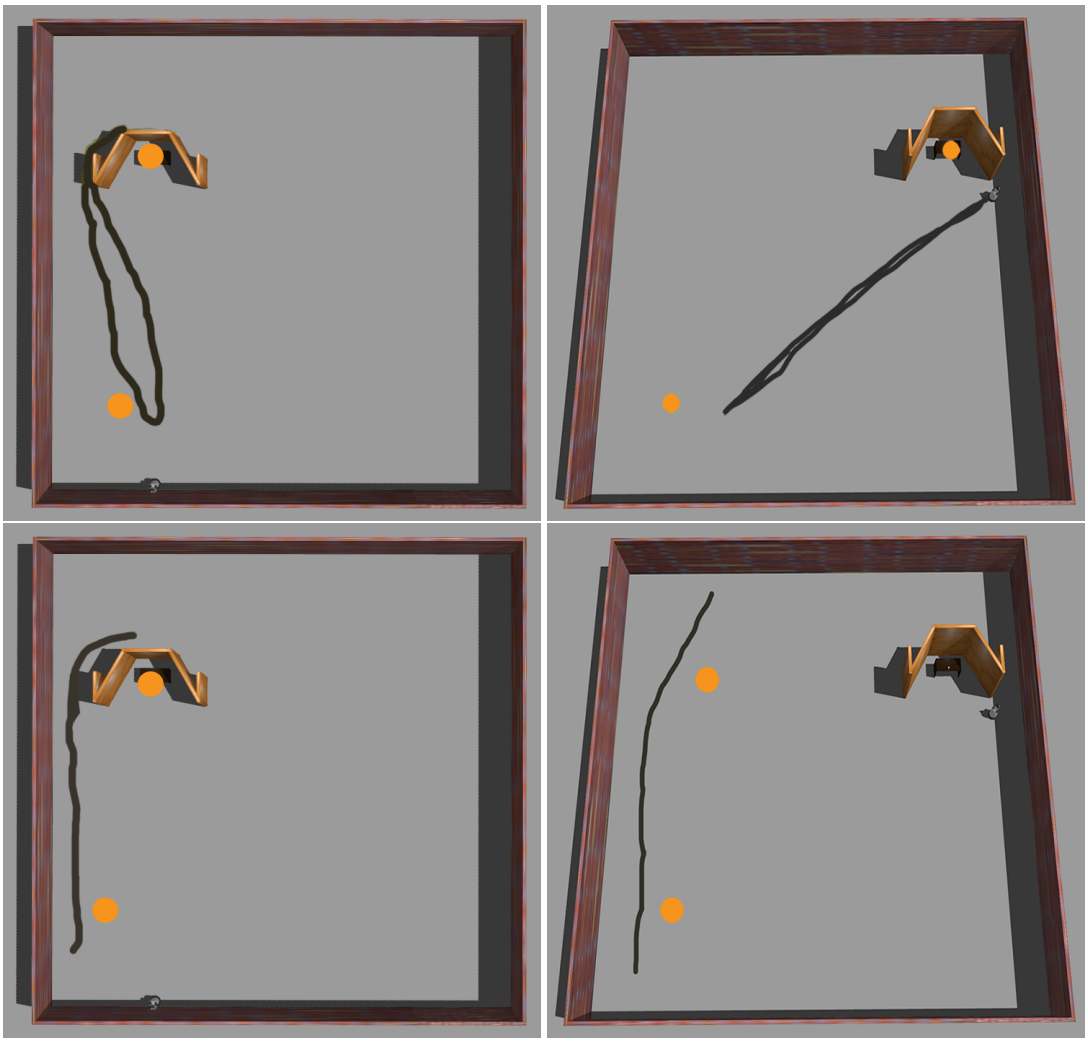


Figure 5: **Top:** Events dependent on furniture. **Bottom:** Events independent of furniture. Black line shows the repeated sweeping path the robot takes. Orange dots are the locations where events fire.

7 References

[Ahmadi and Stone, 2005] Mazda Ahmadi and Peter Stone. Continuous area sweeping: A task definition and initial approach. In *Proceedings of International Conference on Advanced Robotics (ICAR)*, 2005.

[Ahmadi and Stone, 2006] Mazda Ahmadi and Peter Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2006.

[Baykal-Gürsoy, 2010] Melike Baykal-Gürsoy. Semi-markov decision processes. *Wiley Encyclopedia of Operations Research and Management Sciences*, 2010.

[Choset, 2001] Howie Choset. Coverage for robotics - A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 2001.

[Feinberg, 1996] Eugene A Feinberg. On measurability and representation of strategic measures in markov decision processes. *Lecture Notes-Monograph Series*, pages 29–43, 1996.

[Gabriely and Rimon, 2001] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 2001.

[Galceran and Carreras, 2013] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robot. Auton. Syst.*, 61(12):1258–1276, December 2013.

[Hart *et al.*, 2018] Justin W Hart, Rishi Shah, Sean Kirmani, Nick Walker, Kathryn Baldauf, Nathan John, and Peter Stone. PRISM: Pose Registration for Integrated Semantic Mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[Jebari *et al.*, 2011] Islem Jebari, Stéphane Bazeille, Emmanuel Battesti, Hassene Tekaya, Marius Klein, Adriana Tapus, David Filliat, Cédric Meyer, Sio Hoï Ieng, Ryad Benosman, Eddy Cizeron, Jean Charles Mamanna, and Benoit Pothier. Multi-sensor semantic mapping and exploration of indoor environments. In *IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2011.

[Koenig and Howard, 2004] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, 2004.

[Kunze *et al.*, 2018] Lars Kunze, Hakan Karaoguz, Jay Young, Ferdian Jovan, John Folkesson, Patric Jensfelt, and Nick Hawes. SOMA: A framework for understanding change in everyday environments using semantic object maps. In *Proceedings 2018 AAAI Fall Symposium on Reasoning and Learning in Real-World Systems for Long-Term Autonomy*, pages 47–54. AAAI, 2018.

[Kurabayashi *et al.*, 1996] Daisuke Kurabayashi, Jun Ota, Tamio Arai, and Eiichi Yoshida. Cooperative sweeping by multiple mobile robots. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1744–1749. IEEE, 1996.

[Mason and Marthi, 2012] Julian Mason and Bhaskara Marthi. An object-based semantic world model for long-term change detection and semantic querying. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3851–3858. IEEE, 2012.

[Nüchter *et al.*, 2006] Andreas Nüchter, Oliver Wulf, Kai Lingemann, Joachim Hertzberg, Bernardo Wagner, and Hartmut Surmann. 3d mapping with semantic knowledge. In *RoboCup 2005: Robot Soccer World Cup IX*, pages 335–346, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Schwartz, 1993] Anton Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning*, volume 298, pages 298–305, 1993.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[Yamauchi, 1997] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, July 1997.

[Zlot *et al.*, 2002] Robert Zlot, Anthony Stentz, M Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 3, pages 3016–3023. IEEE, 2002.